

1994 Shell Caribbean Cup

Barbados

Grenada

Needs to win match by two goals

Needs to win match or lose by one goal

Tournament Rule: Goals in overtime count double

Late in the game Barbados leads 2-0, begins playing defensively

83rd minute : Grenada scores, now 2-1

87th minute : Still 2-1

88th minute : Barbados scores own goal, now 2-2

Grenada realizes that if it scores in *either goal*, it will advance in the tournament

Barbados successfully defends both goals and wins in overtime

But loses in the next round.

Learning

CS 121

Lecture 12

August 3, 2009

R&N Chapter 18

Chris Archibald

Today : Learning

- Why learning?
- Basic Ideas
- Decision Trees

What is Learning?

- Mostly generalization from experience:
- “Our experience of the world is specific, yet we are able to formulate general theories that account for the past and predict the future”
M.R. Genesereth and N.J. Nilsson,
in *Logical Foundations of AI*, 1987
- Concepts, heuristics, policies

Learning

- Learning is essential for unknown environments
 - i.e. when designer lacks omniscience
- Learning is useful as system construction method
 - i.e. expose agent to reality rather than try to write it down
- Learning modifies the agent's decision mechanisms to improve performance

Learning

- Main idea: agents should use their percepts not only for acting, but also for improving their future ability to act
- Wide range of methods
- Major design issue is the type of *feedback* that will be available to the agent

Types of learning settings

- Supervised
 - Given a set of example inputs and outputs
 - Goal is to learn a function relating the two
- Unsupervised
 - Given inputs, but no outputs
 - Goal is to group input into different classes
- Reinforcement
 - Agent given rewards in certain states (think MDPs)
 - Goal is to learn a policy that will maximize rewards

Examples

- Supervised learning
 - Taxi learning to brake w/ instructor
 - Spam filter
- Unsupervised learning
 - Market research
 - Data mining
- Reinforcement learning
 - Helicopter flying (videos we've seen)
 - MDPs

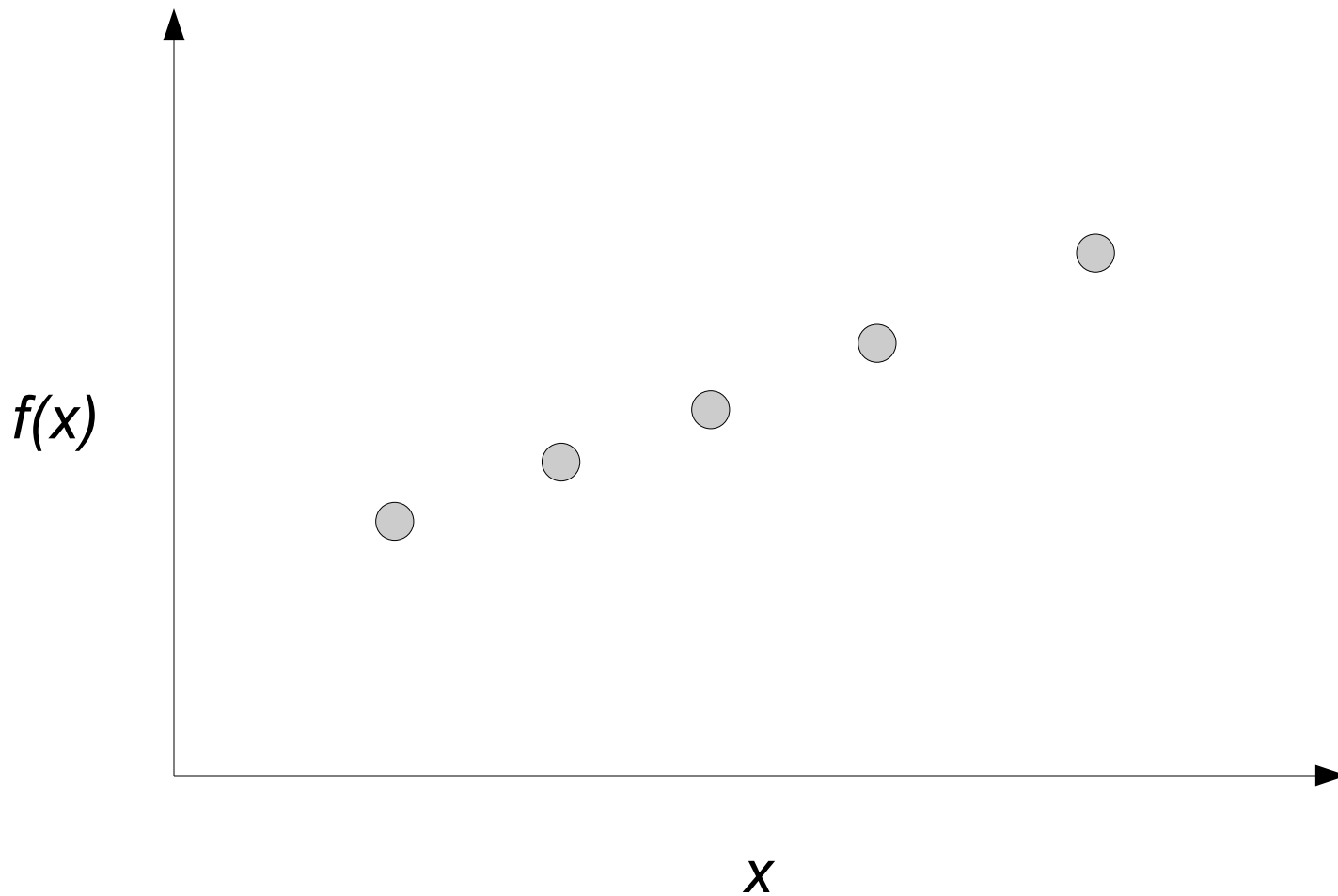
Other factors

- Representation of learned information
- Availability of prior knowledge

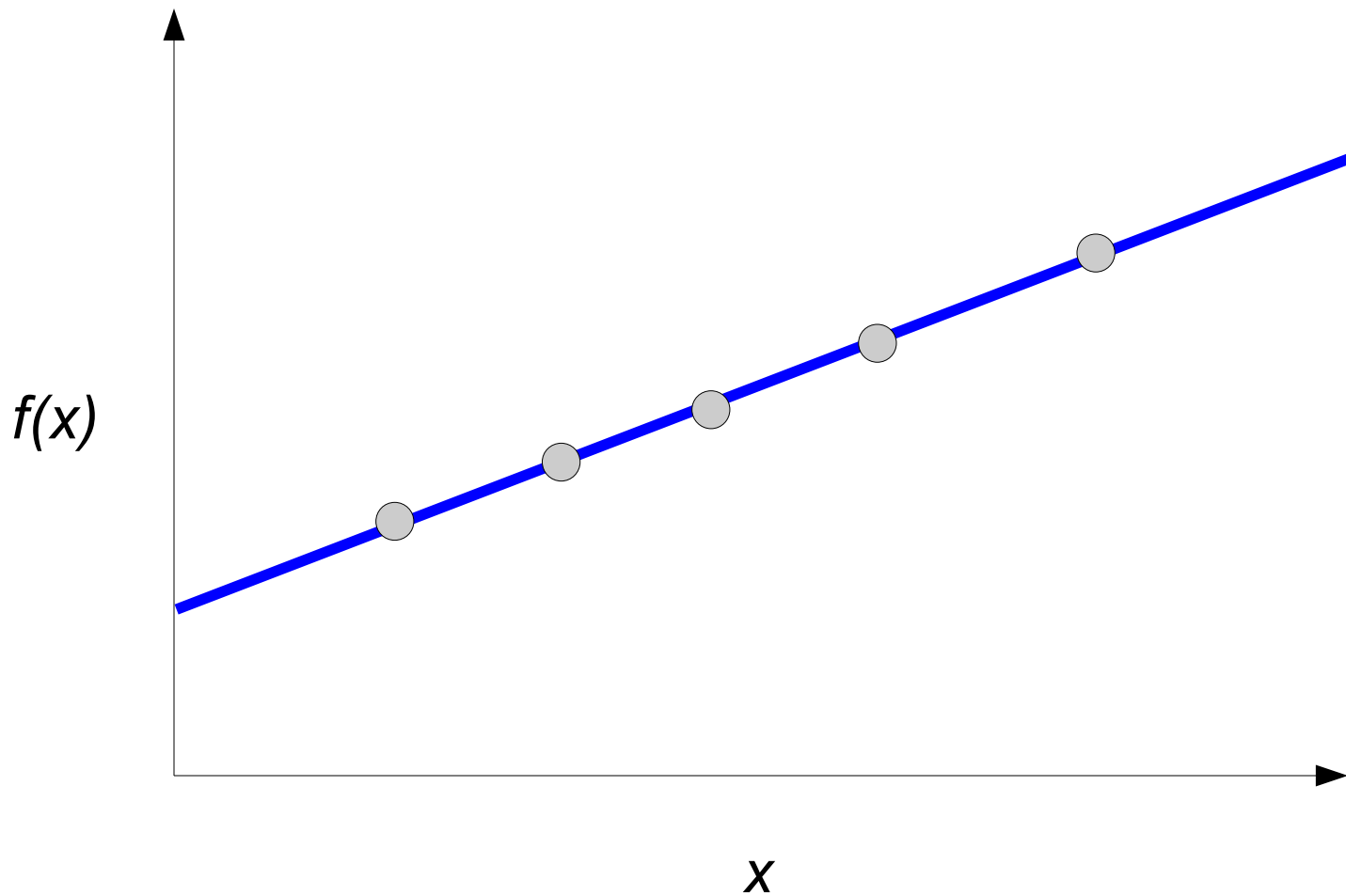
Inductive learning

- Let's call an *example* a pair $(x, f(x))$, where x is the input and $f(x)$ is the output of the function applied to x
- Pure inductive inference:
 - Given a collection of examples of f , return a function h that approximates f
 - h is called a *hypothesis*
- How can we tell if a hypothesis is good?

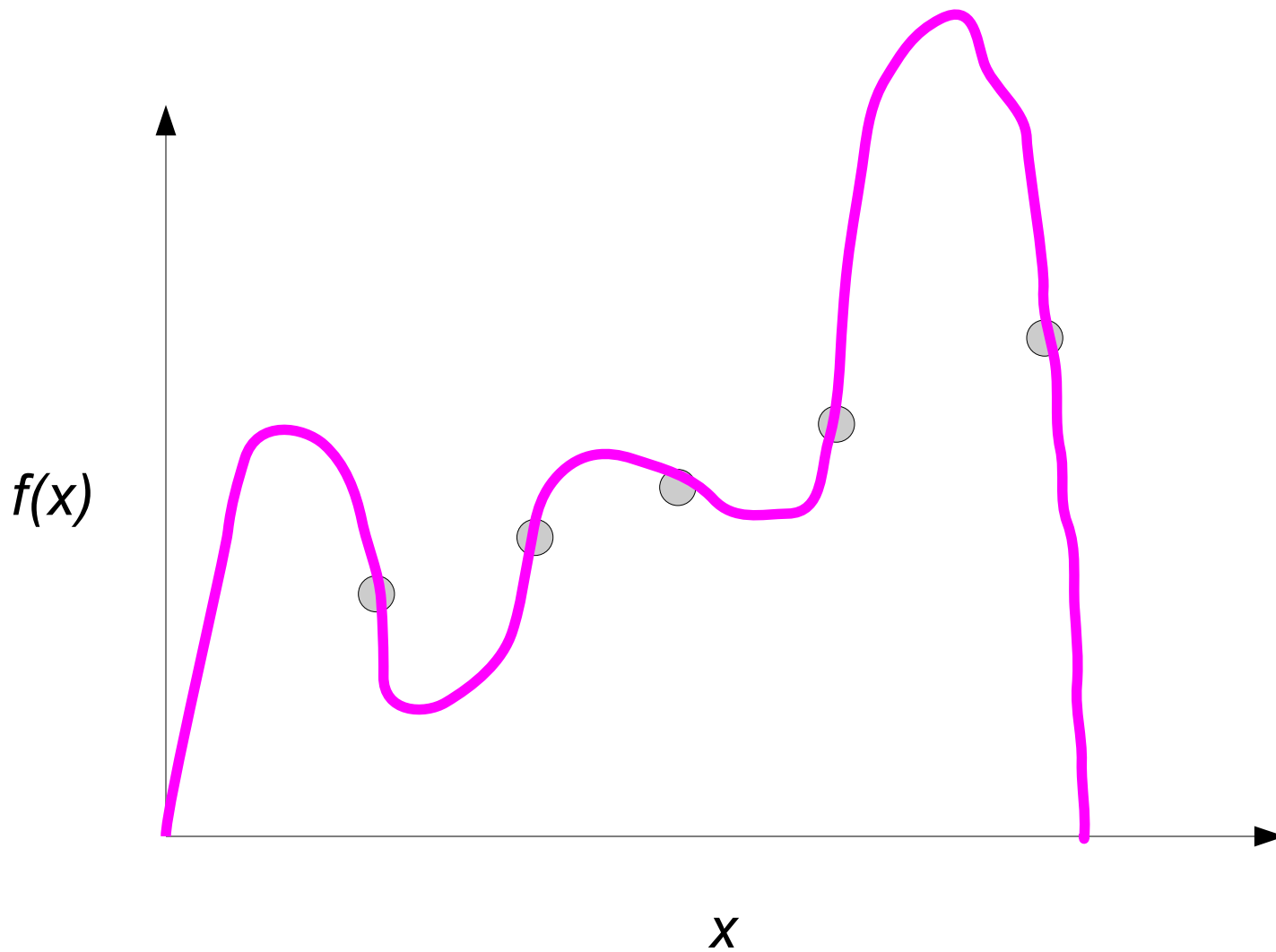
Example



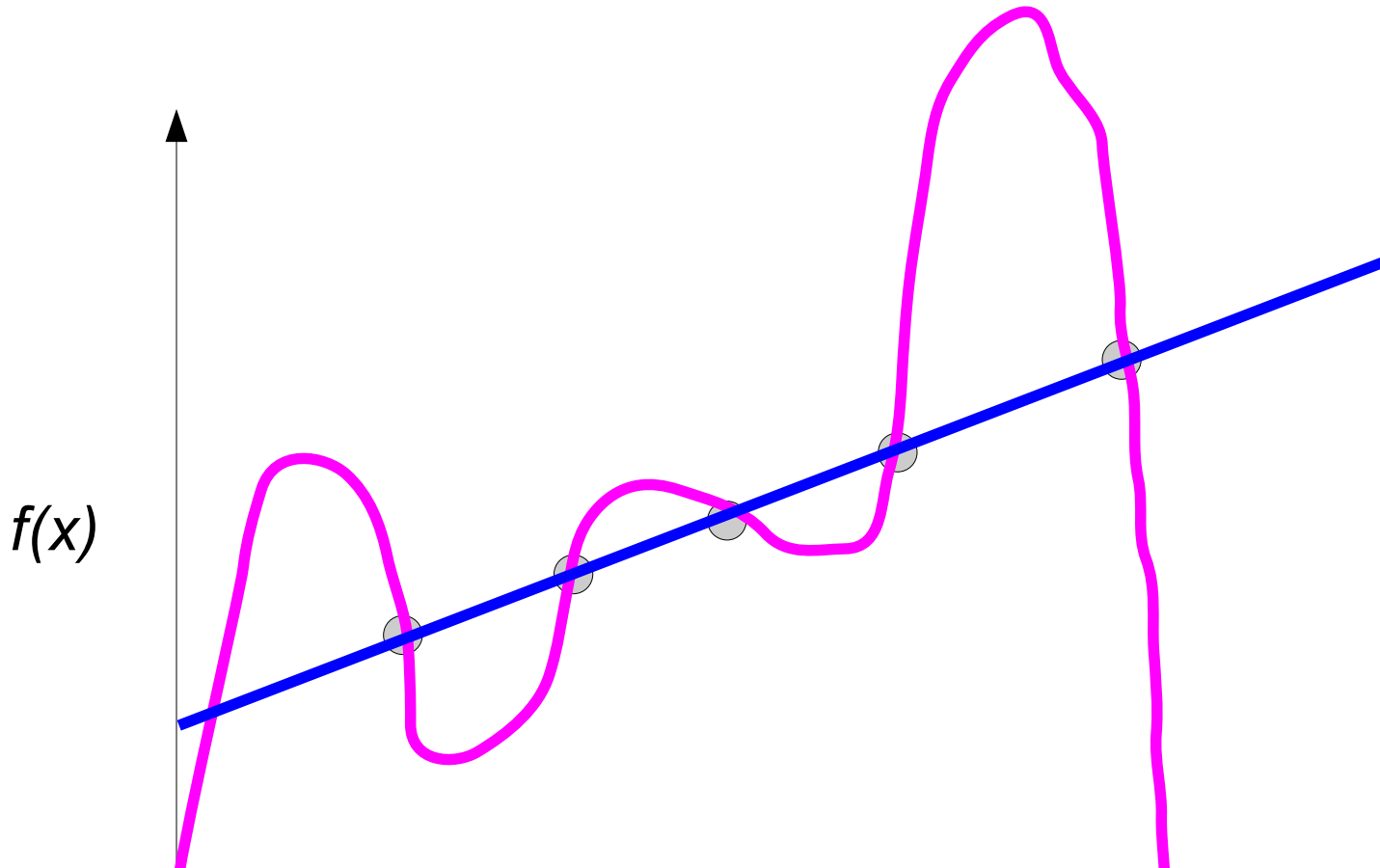
Example



Example



Example



How do we decide between these two hypotheses that both agree with our data?

What we desire from a hypothesis

- Since we will use the hypothesis h most often to predict the output of $f(x)$ on examples we haven't seen yet, we want it to do well on these
- We call this *generalization*
- Ideally we would like to find an h such that

$$h = f$$

Tradeoff: complexity vs data-fit

- A *hypothesis space* is the set of all functions that we consider as a hypothesis
- Generally, the larger and more complex the hypothesis space is, the better we can fit our data
- If f is in our hypothesis space, we say that the learning problem is *realizable*
- Otherwise, it is *unrealizable*
- Prior knowledge can help us find right hypothesis space

Tradeoff: complexity vs. data-fit

- Why not use something super general, like let H be the space of all Turing machines?
- We need to take into account the *computational complexity* of learning
 - Fitting straight lines = easy
 - Fitting high-degree polynomials = harder
 - Fitting Turing machines = undecidable
- Also want to take into account how hard it is to use h . Prefer fast computation time
- Learning typically focuses on simple representations

Contents

- Introduction to inductive learning
- Logic-based inductive learning:
 - Decision-tree induction
- In future: Function-based inductive learning
 - Neural nets

Logic-Based Inductive Learning

- **Background** knowledge KB
- Training set D (**observed** knowledge) that is not logically implied by KB
- **Inductive inference:**
Find h such that KB and h imply D

$h = D$ is a trivial, but un-interesting solution (data caching)

Rewarded Card Example

- Deck of cards, with each card designated by $[r,s]$, its rank and suit, and some cards "rewarded"
- **Background knowledge KB:**
 - $((r=1) \vee \dots \vee (r=10)) \Leftrightarrow \text{NUM}(r)$
 - $((r=J) \vee (r=Q) \vee (r=K)) \Leftrightarrow \text{FACE}(r)$
 - $((s=S) \vee (s=C)) \Leftrightarrow \text{BLACK}(s)$
 - $((s=D) \vee (s=H)) \Leftrightarrow \text{RED}(s)$
- **Training set D:**
 - $\text{REWARD}([4,C]) \wedge \text{REWARD}([7,C]) \wedge \text{REWARD}([2,S]) \wedge$
 $\neg \text{REWARD}([5,H]) \wedge \neg \text{REWARD}([J,S])$

Rewarded Card Example

- Deck of cards, with each card designated by $[r,s]$, its rank and suit, and some cards "rewarded"

- **Background knowledge KB:**

$$((r=1) \vee \dots \vee (r=10)) \Leftrightarrow \text{NUM}(r)$$

$$((r=J) \vee (r=Q) \vee (r=K)) \Leftrightarrow \text{FACE}(r)$$

$$((s=S) \vee (s=C)) \Leftrightarrow \text{BLACK}(s)$$

$$((s=D) \vee (s=H)) \Leftrightarrow \text{RED}(s)$$

There are several possible inductive hypotheses

- **Training set D:**

$$\text{REWARD}([4,C]) \wedge \text{REWARD}([7,C]) \wedge \text{REWARD}([2,S]) \wedge \\ \neg \text{REWARD}([5,H]) \wedge \neg \text{REWARD}([J,S])$$

- **Possible inductive hypothesis:**

$$h \equiv (\text{NUM}(r) \wedge \text{BLACK}(s) \Leftrightarrow \text{REWARD}([r,s]))$$

Learning a Predicate (Concept Classifier)

- Set E of objects (e.g., cards)
- **Goal** predicate $\text{CONCEPT}(x)$, where x is an object in E , that takes the value True or False (e.g., REWARD)

Example:

CONCEPT describes the precondition of an action, e.g.,

Unstack(C,A)

- E is the set of states
- **CONCEPT**(x) \Leftrightarrow

HANDEEMPTY $\in x$, **BLOCK**(C) $\in x$, **BLOCK**(A) $\in x$,

CLEAR(C) $\in x$, **ON**(C,A) $\in x$

Learning **CONCEPT** is a step toward learning an action description

Learning a Predicate (Concept Classifier)

- Set E of objects (e.g., cards)
- Goal predicate $\text{CONCEPT}(x)$, where x is an object in E , that takes the value True or False (e.g., REWARD)
- **Observable** predicates $A(x)$, $B(x)$, ... (e.g., NUM, RED)
- **Training set**: values of CONCEPT for some combinations of values of the observable predicates

Example of Training Set

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Example of Training Set

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Hot	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Sunny	Hot	High	Weak	Yes
D8	Sunny	Hot	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes

Ternary attributes

Goal predicate is PLAY-TENNIS

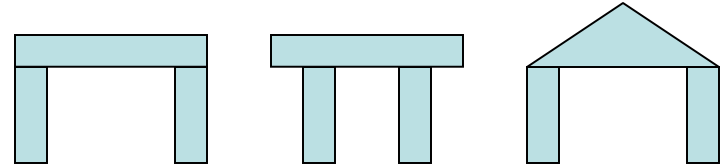
Note that the training set does not say whether an observable predicate is pertinent or not

Learning a Predicate (Concept Classifier)

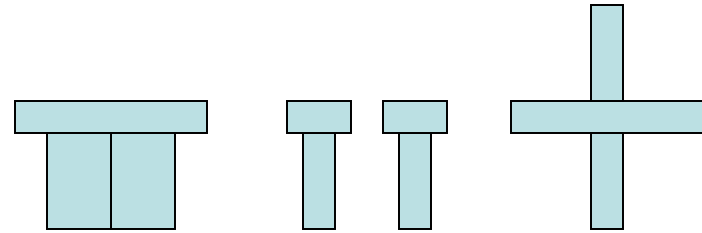
- Set E of objects (e.g., cards)
- Goal predicate $\text{CONCEPT}(x)$, where x is an object in E , that takes the value True or False (e.g., REWARD)
- Observable predicates $A(x), B(x), \dots$ (e.g., NUM, RED)
- Training set: values of CONCEPT for some combinations of values of the observable predicates
- Find a representation of CONCEPT in the form:
$$\text{CONCEPT}(x) \Leftrightarrow S(A, B, \dots)$$
where $S(A, B, \dots)$ is a sentence built with the observable predicates, e.g.:
$$\text{CONCEPT}(x) \Leftrightarrow A(x) \wedge (\neg B(x) \vee C(x))$$

Learning an Arch Classifier

- These objects are arches:
(positive examples)



- These aren't:
(negative examples)



$$\begin{aligned} \text{ARCH}(x) \Leftrightarrow & \text{HAS-PART}(x,b1) \wedge \text{HAS-PART}(x,b2) \wedge \\ & \text{HAS-PART}(x,b3) \wedge \text{IS-A}(b1,\text{BRICK}) \wedge \\ & \text{IS-A}(b2,\text{BRICK}) \wedge \neg \text{MEET}(b1,b2) \wedge \\ & (\text{IS-A}(b3,\text{BRICK}) \vee \text{IS-A}(b3,\text{WEDGE})) \wedge \\ & \text{SUPPORTED}(b3,b1) \wedge \text{SUPPORTED}(b3,b2) \end{aligned}$$

Example set

- An **example** consists of the values of CONCEPT and the observable predicates for some object x
- A example is **positive** if CONCEPT is True, else it is **negative**
- The set X of all examples is the **example set**
- The **training set** is a **subset** of X



a small one!

Hypothesis Space

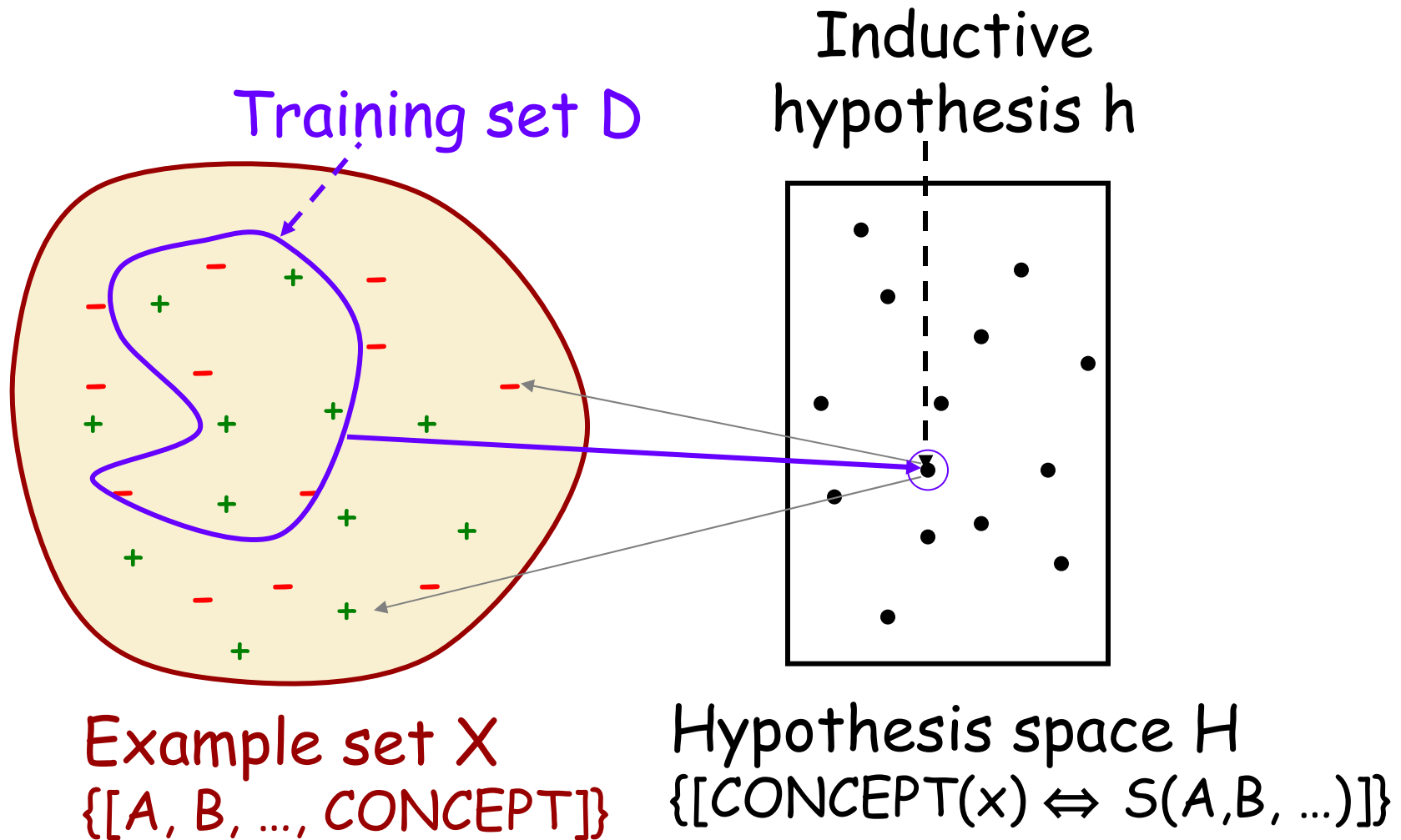
- An **hypothesis** is any sentence of the form:

$$\text{CONCEPT}(x) \Leftrightarrow S(A, B, \dots)$$

where $S(A, B, \dots)$ is a sentence built using the observable predicates

- The set of all hypotheses is called the **hypothesis space** H
- An hypothesis h **agrees** with an example if it gives the correct value of **CONCEPT**

Inductive Learning Scheme



Size of Hypothesis Space

- n observable predicates
- 2^n entries in truth table defining CONCEPT and each entry can be filled with True or False
- In the absence of any restriction (bias), there are 2^{2^n} hypotheses to choose from
- $n = 6 \rightarrow 2 \times 10^{19}$ hypotheses!

Multiple Inductive Hypotheses

- Deck of cards, with each card designated by $[r,s]$, its rank and suit, and some cards "rewarded"
- **Background knowledge KB:**
 $((r=1) \vee \dots \vee (r=10)) \Leftrightarrow \text{NUM}(r)$
 $((r=J) \vee (r=Q) \vee (r=K)) \Leftrightarrow \text{FACE}(r)$
 $((s=S) \vee (s=C)) \Leftrightarrow \text{BLACK}(s)$
 $((s=D) \vee (s=H)) \Leftrightarrow \text{RED}(s)$
- **Training set D:**
 $\text{REWARD}([4,C]) \wedge \text{REWARD}([7,C]) \wedge \text{REWARD}([2,S]) \wedge$
 $\neg \text{REWARD}([5,H]) \wedge \neg \text{REWARD}([J,S])$

$$h_1 \equiv \text{NUM}(r) \wedge \text{BLACK}(s) \Leftrightarrow \text{REWARD}([r,s])$$

$$h_2 \equiv \text{BLACK}(s) \wedge \neg(r=J) \Leftrightarrow \text{REWARD}([r,s])$$

$$h_3 \equiv ([r,s]=[4,C]) \vee ([r,s]=[7,C]) \vee [r,s]=[2,S] \\ \Leftrightarrow \text{REWARD}([r,s])$$

$$h_4 \equiv \neg([r,s]=[5,H]) \vee \neg([r,s]=[J,S]) \Leftrightarrow \text{REWARD}([r,s])$$

agree with all the examples in the training set

Multiple Inductive Hypotheses

- Deck of cards, with each card designated by $[r,s]$, its rank and suit, and some cards "rewarded"

Need for a system of preferences - called a bias - to compare possible hypotheses

$$((s=D) \vee (s=H)) \Leftrightarrow \text{RED}(s)$$

- Training set D:

$$\text{REWARD}([4,C]) \wedge \text{REWARD}([7,C]) \wedge \text{REWARD}([2,S]) \wedge \\ \neg \text{REWARD}([5,H]) \wedge \neg \text{REWARD}([J,S])$$

$$h_1 \equiv \text{NUM}(r) \wedge \text{BLACK}(s) \Leftrightarrow \text{REWARD}([r,s])$$

$$h_2 \equiv \text{BLACK}(s) \wedge \neg(r=J) \Leftrightarrow \text{REWARD}([r,s])$$

$$h_3 \equiv ([r,s]=[4,C]) \vee ([r,s]=[7,C]) \vee [r,s]=[2,S] \\ \Leftrightarrow \text{REWARD}([r,s])$$

$$h_4 \equiv \neg([r,s]=[5,H]) \vee \neg([r,s]=[J,S]) \Leftrightarrow \text{REWARD}([r,s])$$

agree with all the examples in the training set

Notion of Capacity

- It refers to the ability of an agent to learn any training set without error
- An agent with too much capacity is like a botanist with photographic memory who, when presented with a new tree, concludes that it is not a tree because it has a different number of leaves from anything he has seen before
- An agent with too little capacity is like the botanist's lazy brother, who declares that if it's green, it's a tree
- Good generalization can only be achieved when the right balance is struck between the accuracy attained on the training set and the capacity of the agent

→ Keep-It-Simple (KIS) Bias

▪ Examples

- Use much fewer observable predicates than the training set
- Constrain the learnt predicate, e.g., to use only “high-level” observable predicates such as NUM, FACE, BLACK, and RED and/or to have simple syntax

▪ Motivation

- If an hypothesis is too complex it is not worth learning it (data caching does the job as well)
- There are much fewer simple hypotheses than complex ones, hence the hypothesis space is smaller

→ Keep-It-Simple (KIS) Bias

■ Examples

- Use much fewer observable predicates than the training set
- Constrain the learnt predicate, e.g., to use only “high-level” observable predicates such as NUM, FACE, BLACK, and RED and/or to have simple syntax

■ Einstein: “A theory must be as simple as possible, but not simpler than this”

- If an hypothesis is too complex it is not worth learning it (data caching does the job as well)
- There are much fewer simple hypotheses than complex ones, hence the hypothesis space is smaller

→ Keep-It-Simple (KIS) Bias

▪ Examples

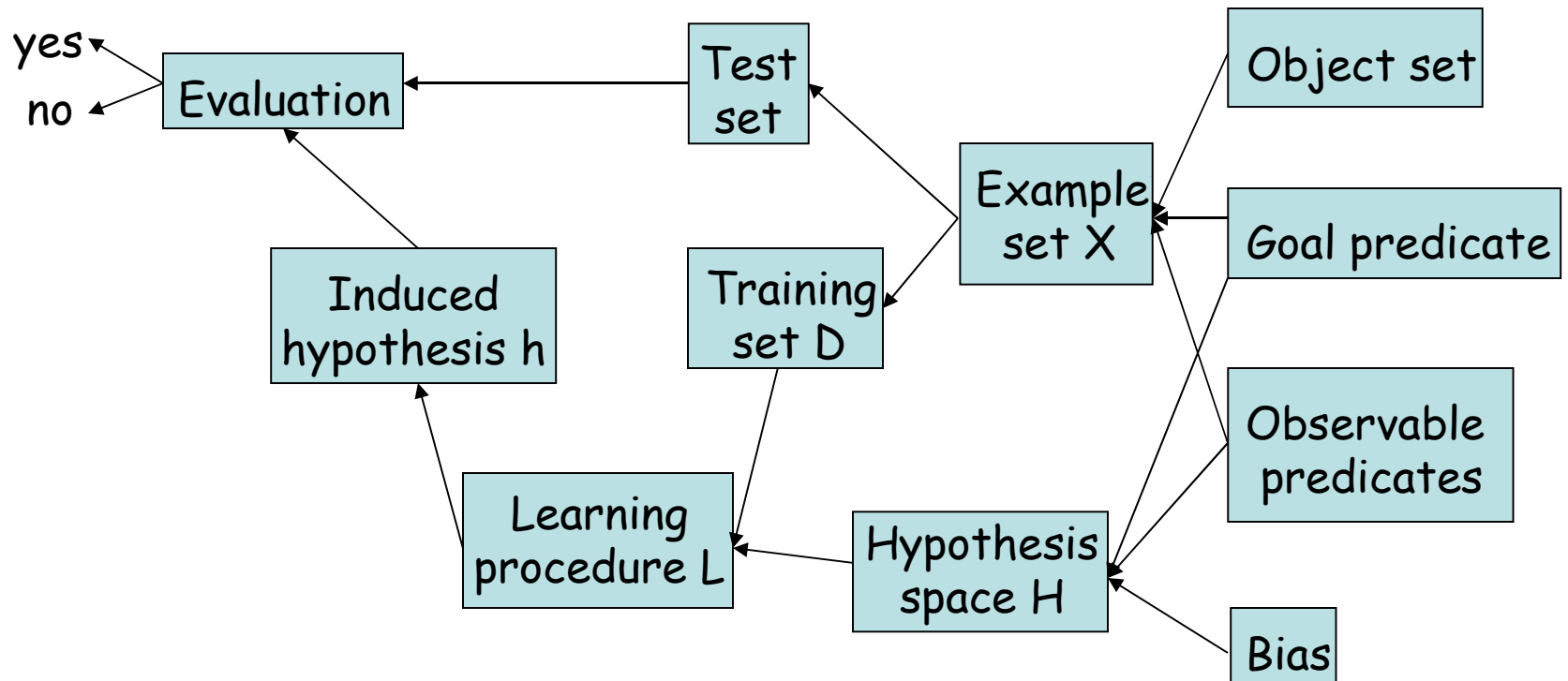
- Use much fewer observable predicates than the training set

If the bias allows only sentences S that are conjunctions of $k \ll n$ predicates picked from the n observable predicates, then the size of H is $O(n^k)$

▪ Motivation

- If an hypothesis is too complex it is not worth learning it (data caching does the job as well)
- There are much fewer simple hypotheses than complex ones, hence the hypothesis space is smaller

Putting Things Together



Decision Tree Method

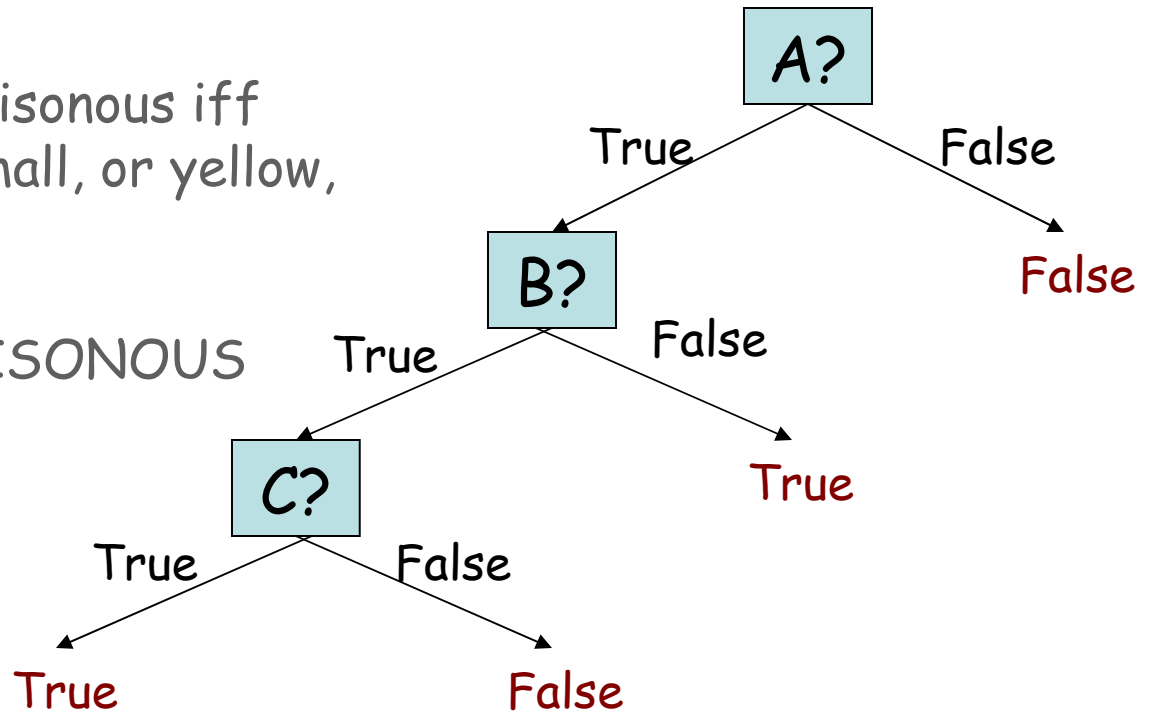
Predicate as a Decision Tree

The predicate $\text{CONCEPT}(x) \Leftrightarrow A(x) \wedge (\neg B(x) \vee C(x))$ can be represented by the following decision tree:

Example:

A mushroom is poisonous iff it is yellow and small, or yellow, big and spotted

- x is a mushroom
- $\text{CONCEPT} = \text{POISONOUS}$
- $A = \text{YELLOW}$
- $B = \text{BIG}$
- $C = \text{SPOTTED}$



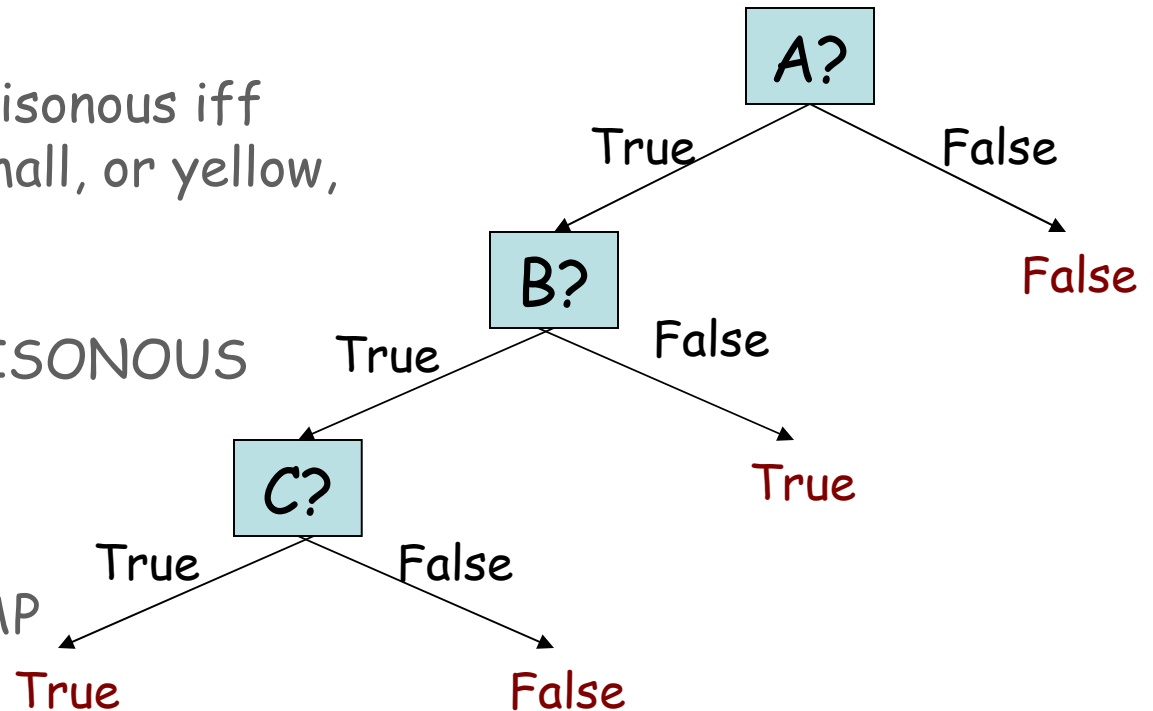
Predicate as a Decision Tree

The predicate $\text{CONCEPT}(x) \Leftrightarrow A(x) \wedge (\neg B(x) \vee C(x))$ can be represented by the following decision tree:

Example:

A mushroom is poisonous iff it is yellow and small, or yellow, big and spotted

- x is a mushroom
- $\text{CONCEPT} = \text{POISONOUS}$
- $A = \text{YELLOW}$
- $B = \text{BIG}$
- $C = \text{SPOTTED}$
- $D = \text{FUNNEL-CAP}$
- $E = \text{BULKY}$

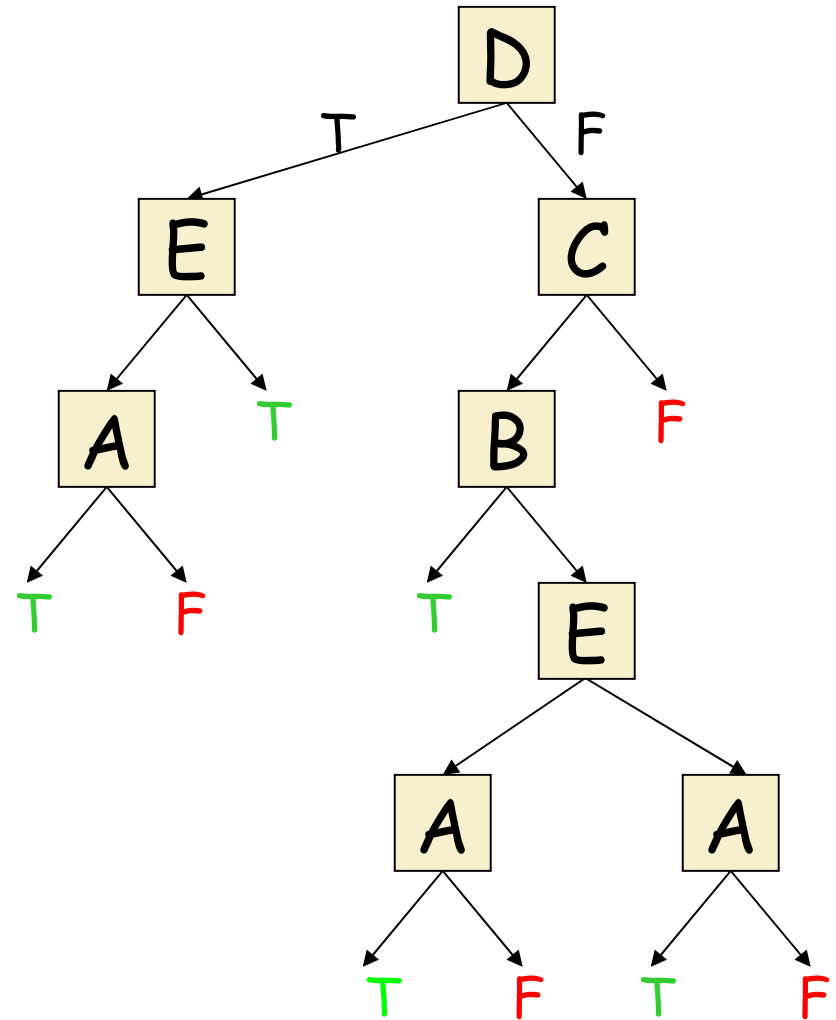


Training Set

Ex. #	A	B	C	D	E	CONCEPT
1	False	False	True	False	True	False
2	False	True	False	False	False	False
3	False	True	True	True	True	False
4	False	False	True	False	False	False
5	False	False	False	True	True	False
6	True	False	True	False	False	True
7	True	False	False	True	False	True
8	True	False	True	False	True	True
9	True	True	True	False	True	True
10	True	True	True	True	True	True
11	True	True	False	False	False	False
12	True	True	False	False	True	False
13	True	False	True	True	True	True

Possible Decision Tree

Ex. #	A	B	C	D	E	CONCEPT
1	False	False	True	False	True	False
2	False	True	False	False	False	False
3	False	True	True	True	True	False
4	False	False	True	False	False	False
5	False	False	False	True	True	False
6	True	False	True	False	False	True
7	True	False	False	True	False	True
8	True	False	True	False	True	True
9	True	True	True	False	True	True
10	True	True	True	True	True	True
11	True	True	False	False	False	False
12	True	True	False	False	True	False
13	True	False	True	True	True	True

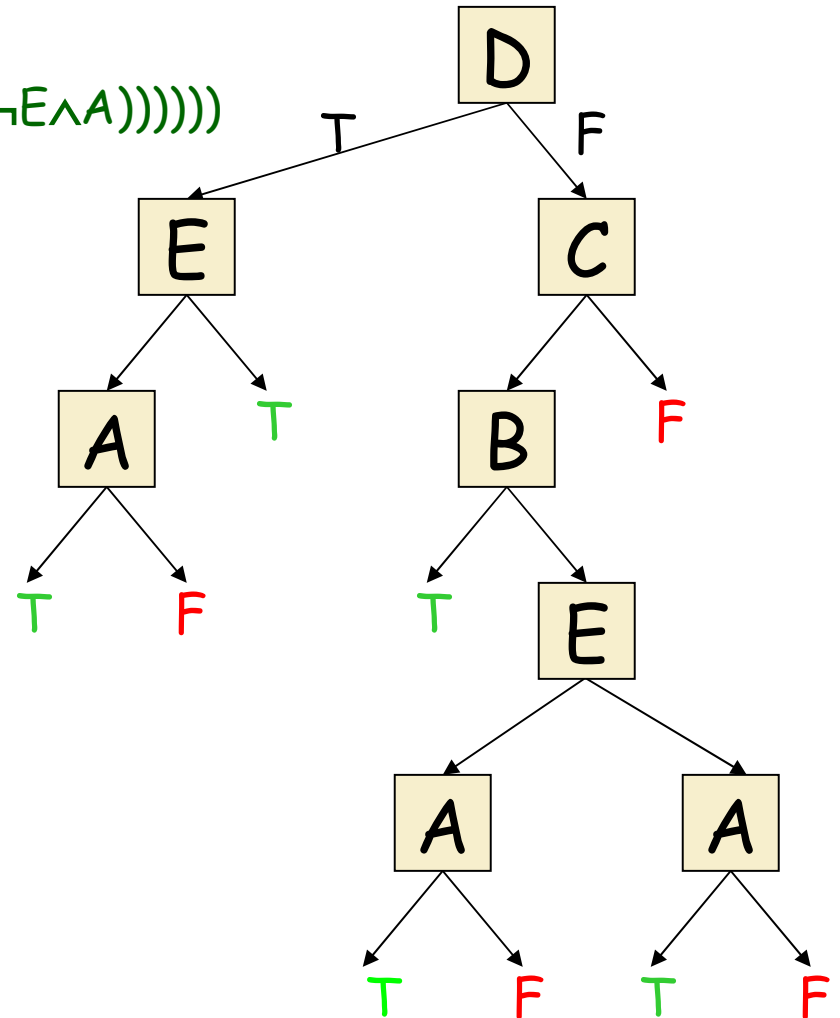
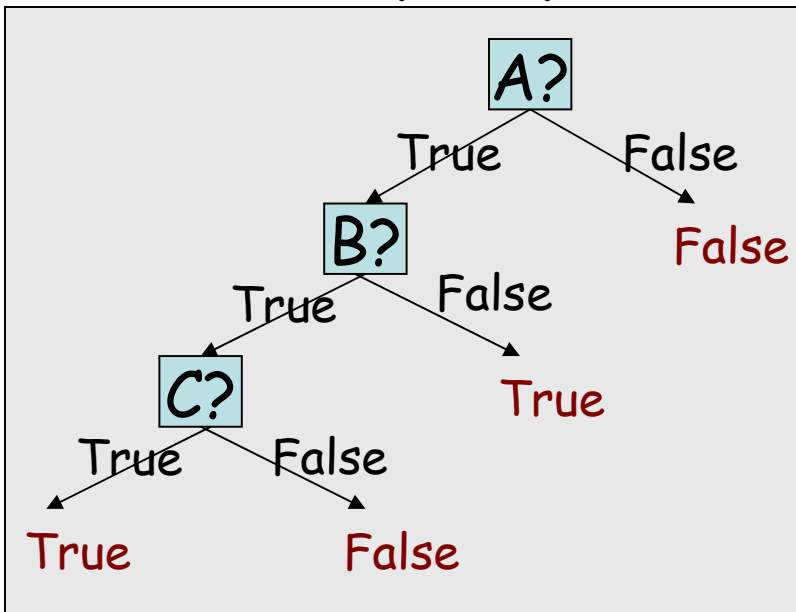


Possible Decision Tree

CONCEPT \Leftrightarrow

$(D \wedge (\neg E \vee A)) \vee (\neg D \wedge (C \wedge (B \vee (\neg B \wedge ((E \wedge A) \vee (\neg E \wedge A))))))$

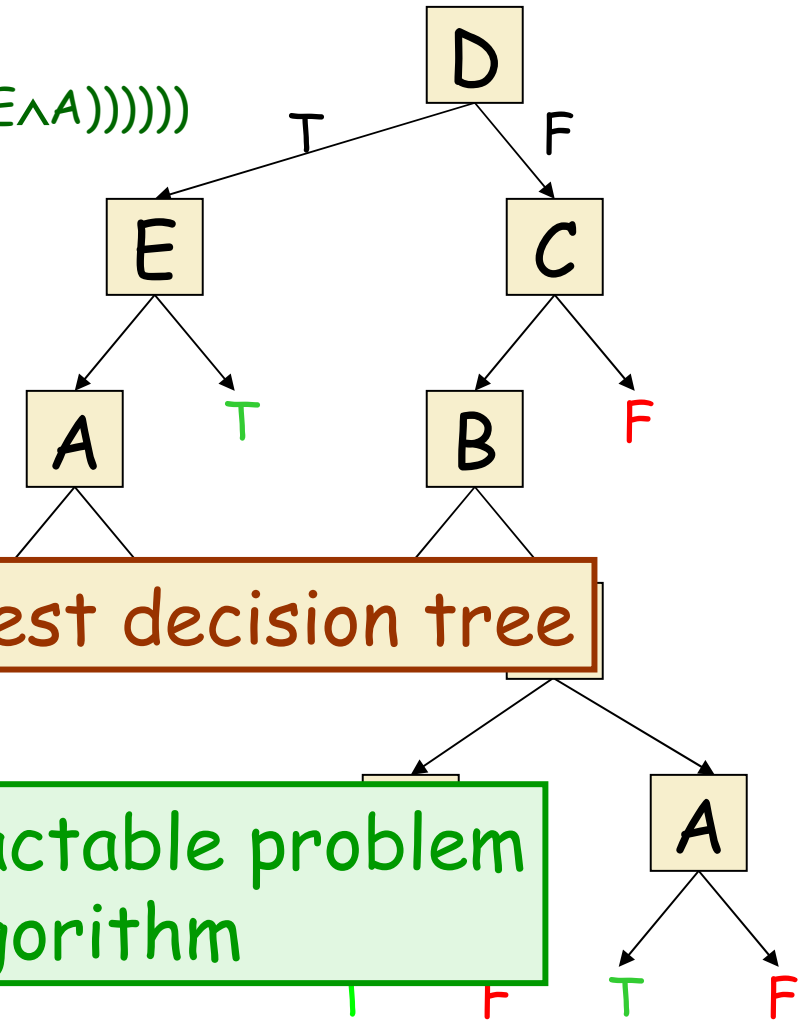
CONCEPT $\Leftrightarrow A \wedge (\neg B \vee C)$



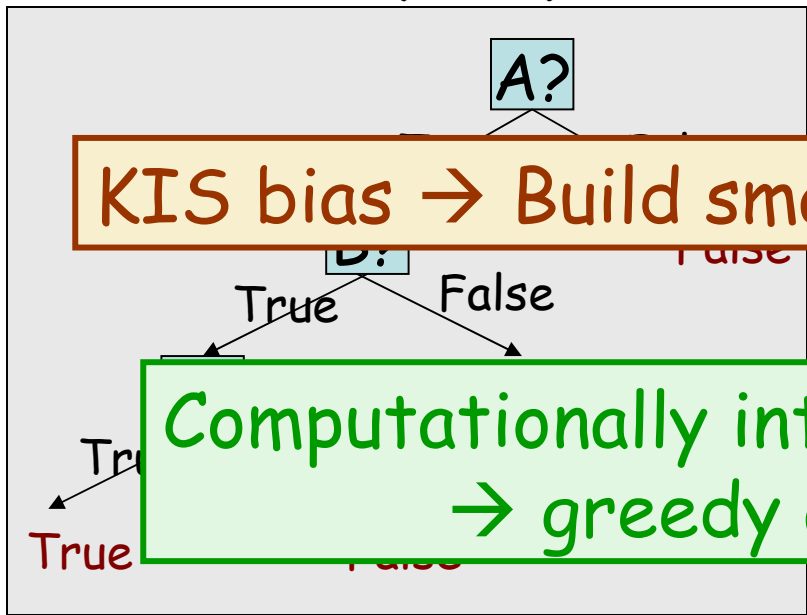
Possible Decision Tree

CONCEPT \Leftrightarrow

$$(D \wedge (\neg E \vee A)) \vee (\neg D \wedge (C \wedge (B \vee (\neg B \wedge ((E \wedge A) \vee (\neg E \wedge A))))))$$



CONCEPT $\Leftrightarrow A \wedge (\neg B \vee C)$



KIS bias \rightarrow Build smallest decision tree

Computationally intractable problem \rightarrow greedy algorithm

Getting Started:

Top-Down Induction of Decision Tree

The distribution of training set is:

True: 6, 7, 8, 9, 10, 13

False: 1, 2, 3, 4, 5, 11, 12

Ex. #	A	B	C	D	E	CONCEPT
1	False	False	True	False	True	False
2	False	True	False	False	False	False
3	False	True	True	True	True	False
4	False	False	True	False	False	False
5	False	False	False	True	True	False
6	True	False	True	False	False	True
7	True	False	False	True	False	True
8	True	False	True	False	True	True
9	True	True	True	False	True	True
10	True	True	True	True	True	True
11	True	True	False	False	False	False
12	True	True	False	False	True	False
13	True	False	True	True	True	True

Getting Started:

Top-Down Induction of Decision Tree

The distribution of training set is:

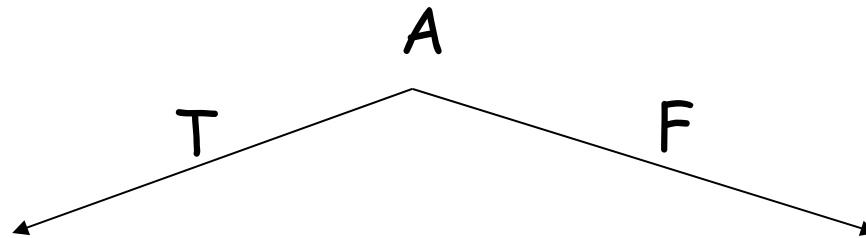
True: 6, 7, 8, 9, 10, 13

False: 1, 2, 3, 4, 5, 11, 12

Without testing any observable predicate, we could report that CONCEPT is False (**majority rule**) with an estimated probability of error $P(E) = 6/13$

Assuming that we will only include one observable predicate in the decision tree, which predicate should we test to minimize the probability of error (i.e., the # of misclassified examples in the training set)? → Greedy algorithm

Assume It's A



True: 6, 7, 8, 9, 10, 13

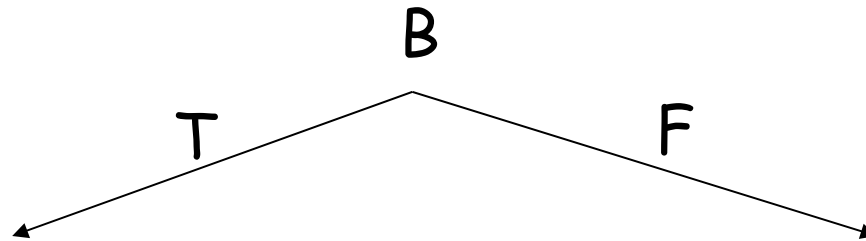
False: 11, 12

1, 2, 3, 4, 5

If we test only A , we will report that CONCEPT is True if A is True (majority rule) and False otherwise

→ The number of misclassified examples from the training set is 2

Assume It's B

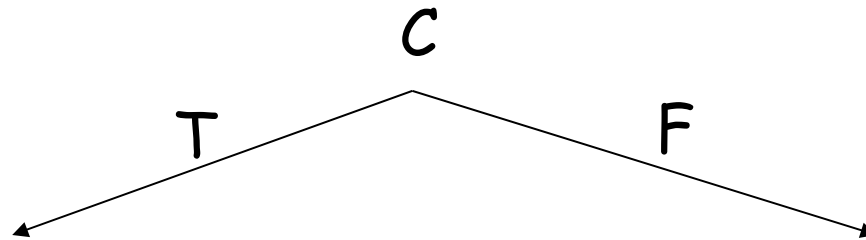


True:	9, 10	6, 7, 8, 13
False:	2, 3, 11, 12	1, 4, 5

If we test only B, we will report that CONCEPT is False if B is True and True otherwise

→ The number of misclassified examples from the training set is 5

Assume It's C

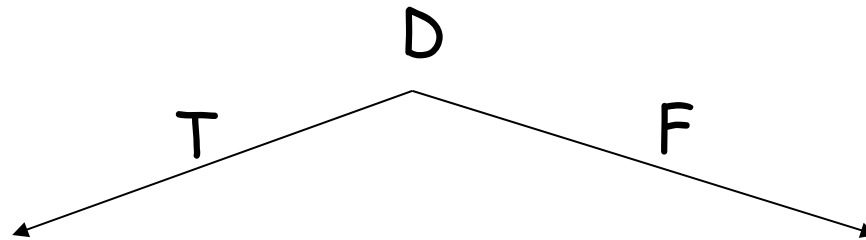


True:	6, 8, 9, 10, 13	7
False:	1, 3, 4	1, 5, 11, 12

If we test only C , we will report that CONCEPT is True if C is True and False otherwise

→ The number of misclassified examples from the training set is 4

Assume It's D



True: 7, 10, 13

6, 8, 9

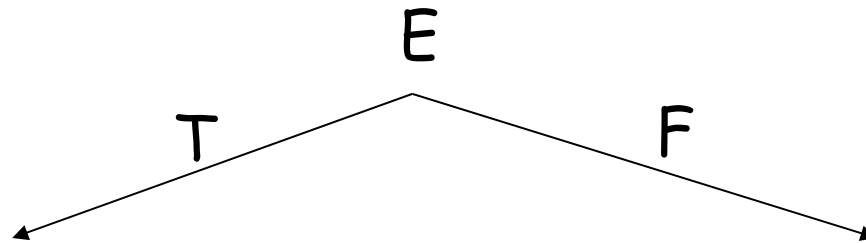
False: 3, 5

1, 2, 4, 11, 12

If we test only D, we will report that CONCEPT is True if D is True and False otherwise

→ The number of misclassified examples from the training set is 5

Assume It's E

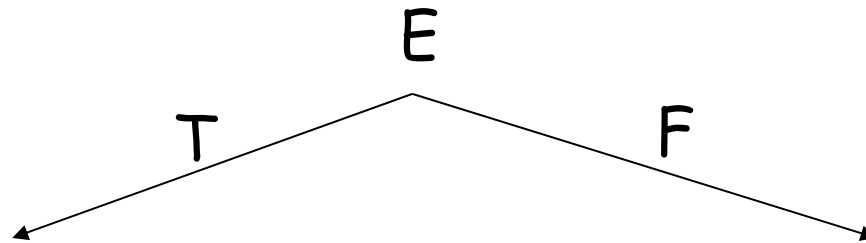


True:	8, 9, 10, 13	6, 7
False:	1, 3, 5, 12	2, 4, 11

If we test only E we will report that CONCEPT is False, independent of the outcome

→ The number of misclassified examples from the training set is 6

Assume It's E

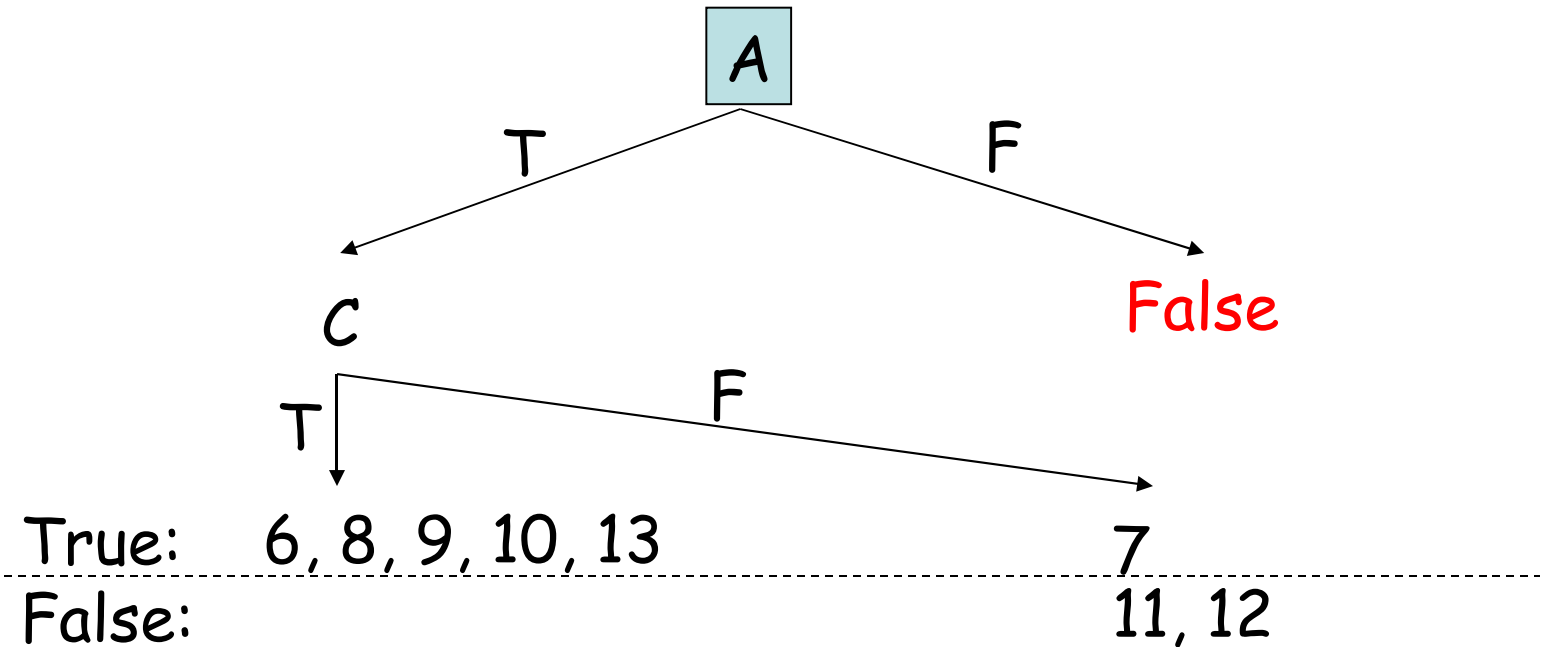


True:	8, 9, 10, 13	6, 7
False:	1, 3, 5, 12	2, 4, 11

So, the best predicate to test is A , independent of the outcome

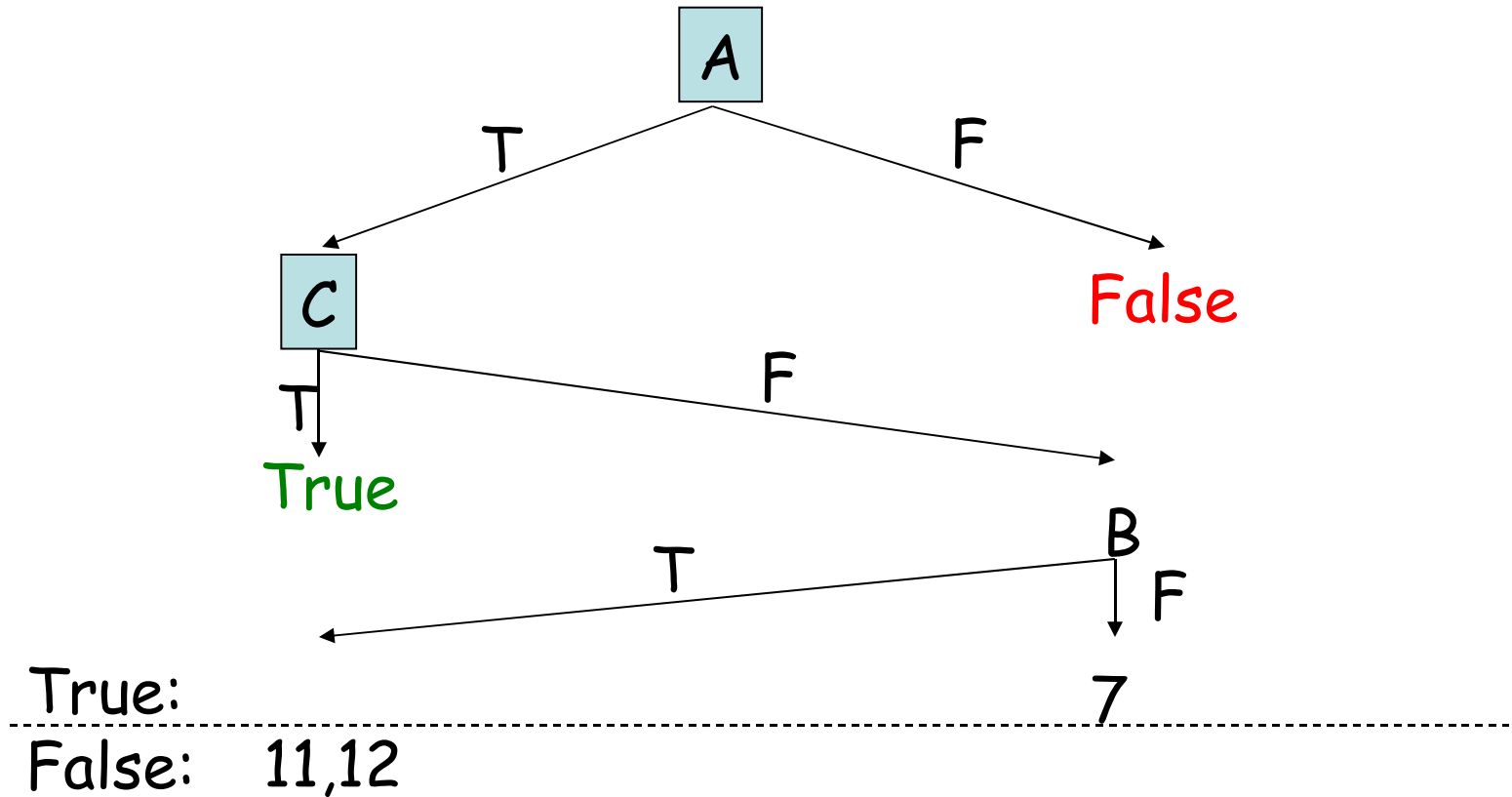
→ The number of misclassified examples from the training set is 6

Choice of Second Predicate

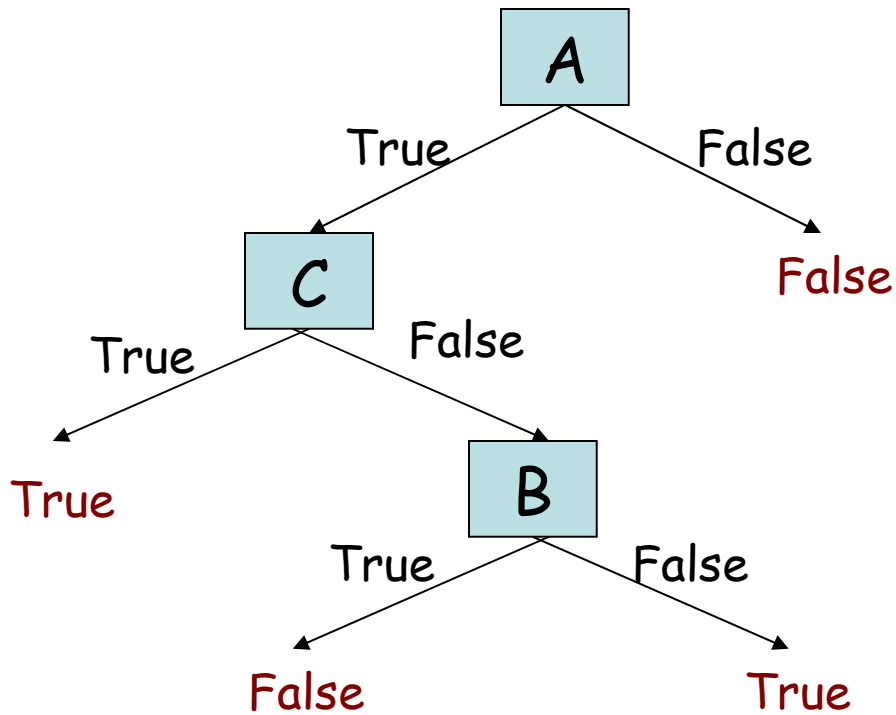


→ The number of misclassified examples from the training set is 1

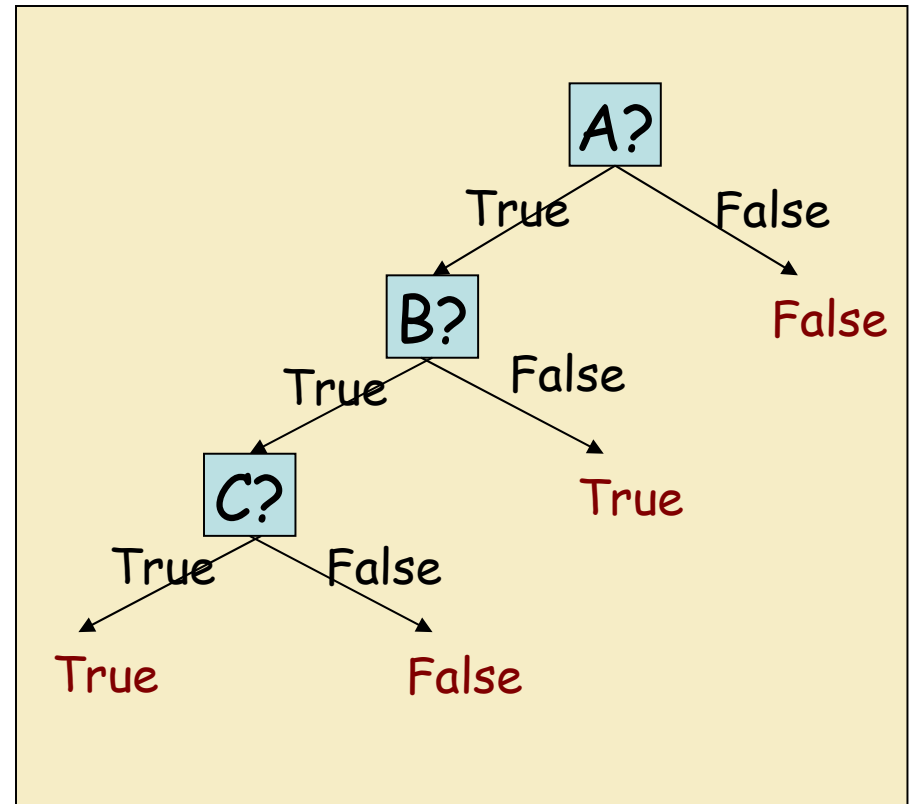
Choice of Third Predicate



Final Tree

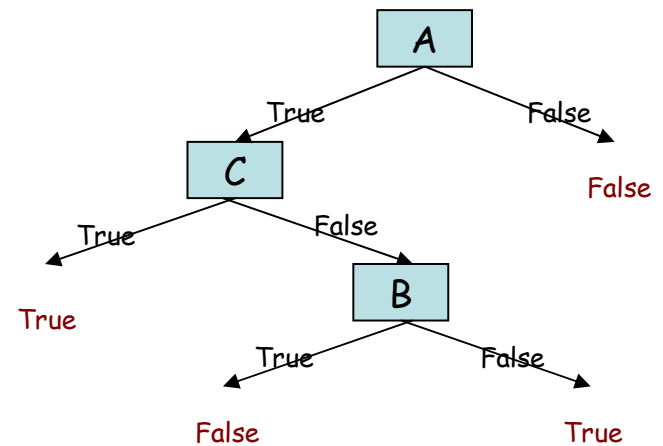


$$\text{CONCEPT} \Leftrightarrow A \wedge (C \vee \neg B)$$



$$\text{CONCEPT} \Leftrightarrow A \wedge (\neg B \vee C)$$

Top-Down Induction of a DT

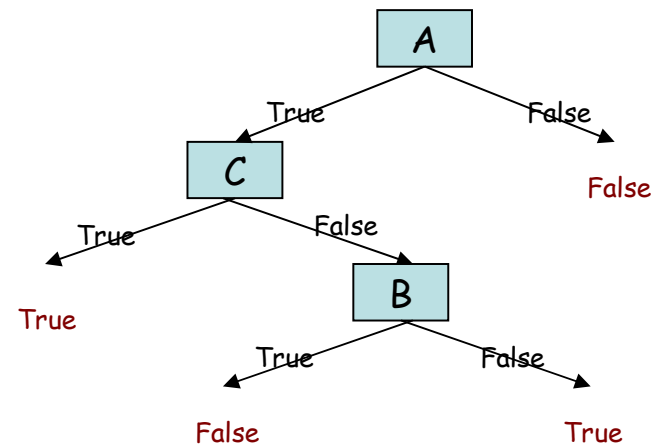


DTL(Δ , *Predicates*)

- ◆ If all examples in Δ are positive then return True
- ◆ If all examples in Δ are negative then return False
- ◆ If *Predicates* is empty then return *failure*
- ◆ $A \leftarrow$ error-minimizing predicate in *Predicates*
- ◆ Return the tree whose:
 - root is A,
 - left branch is DTL(Δ^{+A} , *Predicates*-A),
 - right branch is DTL(Δ^{-A} , *Predicates*-A)

Subset of examples
that satisfy A

Top-Down Induction of a DT



DTL(Δ , *Predicates*)

- ◆ If all examples in Δ are positive then return True
- ◆ If all examples in Δ are negative then return False
- ◆ If *Predicates* is empty then return *failure*
- ◆ $A \leftarrow$ error-minimizing predicate in *Predicates*
- ◆ Return the tree whose:
 - root is A,
 - left branch is DTL(Δ^{+A} , *Predicates*-A),
 - right branch is DTL(Δ^{-A} , *Predicates*-A)

Noise in training set!
May return majority rule,
instead of failure

Subset of examples
that satisfy A

Comments

- Widely used algorithm
- Greedy
- Robust to noise (incorrect examples)
- Not incremental

Using Information Theory

- Rather than minimizing the probability of error, many existing learning procedures **minimize the expected number of questions** needed to decide if an object x satisfies **CONCEPT**
- This minimization is based on a measure of the "quantity of information" contained in the truth value of an observable predicate
- See R&N p. 659-660

Information

Information answers questions

The more clueless I am about the answer initially, the more information is contained in the answer

Scale: 1 bit = answer to Boolean question with prior $\langle 0.5, 0.5 \rangle$

Information in an answer when prior is $\langle P_1, \dots, P_n \rangle$ is

$$H(\langle P_1, \dots, P_n \rangle) = \sum_{i=1}^n -P_i \log_2 P_i$$

(also called entropy of the prior)

Information

Suppose we have p positive and n negative examples at the root

$\Rightarrow H(\langle p/(p+n), n/(p+n) \rangle)$ bits needed to classify a new example

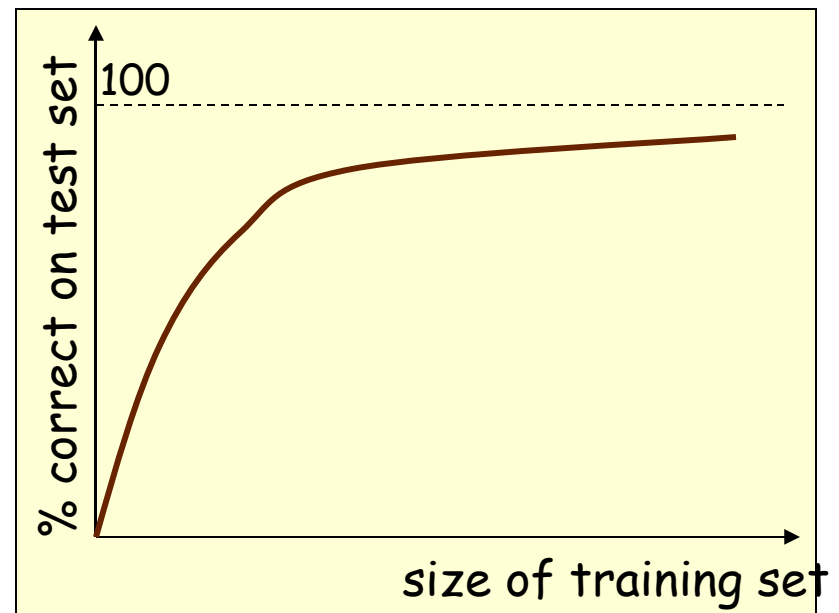
Using Information gain

- Select the attribute that maximizes

$$\textit{Gain}(A) = (\text{Information needed at root}) - (\text{Information needed on average after testing attribute } A)$$

Miscellaneous Issues

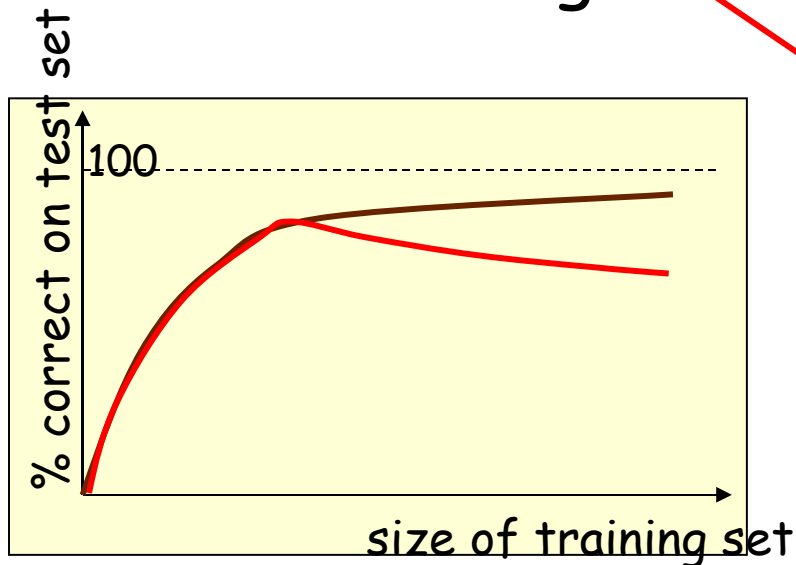
- Assessing performance:
 - Training set and test set
 - Learning curve



Typical learning curve

Miscellaneous Issues

- Assessing performance:
 - Training set and test set
 - Learning curve
- Overfitting



Risk of using irrelevant observable predicates to generate a hypothesis that agrees with all examples in the training set

Miscellaneous Issues

- Assessing performance:
 - Training set and test set
 - Learning curve
- Overfitting
 - Tree pruning

Risk of using irrelevant observable predicates to generate an hypothesis that agrees with all examples in the training set

Terminate recursion when # errors / information gain is small

Miscellaneous Issues

- Assessing performance:
 - Training set and test set
 - Learning curve

- Overfitting

- Tree pruning

Risk of using irrelevant observable predicates to

The resulting decision tree + majority rule may not classify correctly all examples in the training set

sis
mples

Terminate recursion when # errors / information gain is small

Miscellaneous Issues

- Assessing performance:
 - Training set and test set
 - Learning curve
- Overfitting
 - Tree pruning
- Incorrect examples
- Missing data
- Multi-valued and continuous attributes

Applications of Decision Tree

- Medical diagnostic / Drug design
- Evaluation of geological systems for assessing gas and oil basins
- Early detection of problems (e.g., jamming) during oil drilling operations
- Automatic generation of rules in expert systems

Summary

- Learning needed for unknown environments, lazy designers
- Learning method depends on available feedback and representation
- Supervised learning aims to find a simple hypothesis consistent with a set of examples
- Decision tree learning using information gain
- Learning performance = prediction accuracy measured on test set